

VCS Quick Reference

Quick Reference for VCS Module

[PDF Version Here](#)

VCS Module Functions

```
x=vcs.init() # Creates new Canvas
vcs.pause(n) # n seconds pause
vcs.minmax(s) # min,max
vcs.mkevenlevels(n1,n2,nlev=10)
vcs.mkscale(n1,n2,,nc=12,zero=1)
vcs.mlables(vals,output='dict')
vcs.getcolors(levs,colors=range(16,240),split=1,white=240)
vcs.colors.rgb2str(r,g,b) # returns 'name' of closest color from rgb (255 values)
vcs.colors.str2rgb('name') # corresponding r,g,b values
```

Queries

```
vcs.isgraphicmethod() # 1/0
vcs.is[graphicmethod] () #1/0
vcs.istemplate()
vcs.issecondaryobject()
vcs.is[secondaryobject]
vcs.iscolormap()
vcs.isplot()
vcs.graphicsmethodtype(gm)#returns graphic method type (boxfill,...)
```

VCS Canvas (x=vcs.init)

Functions

```
x.open/clear/close()
x.plot(slab,[grid=grd],[graphic method],[template object],...)
x.mode = 1/0 # automatic/manual update
x.update() #updates the changes to vcs canvas
x.flush() #flushes the X server
x.portrait/landscape()
x.page() #toggles between portrait/landscape
x.geometry(width,height,xoffsetleft,yoffsettop) #resize the window
x.set('method',['name']) #set the default graphics method
x.setcolormap('name') #set the colormap
x.setcolorcell(icell,r,g,b) # 0 <= r,g,b <= 100
x.getcolormap('name') #get the colormap 'name'
x.createcolormap('name') #create a colormap named 'name'
x.getcolormapname() #get the name of the colormap used
x.getcolorcell(icell) #return the [r,g,b] values for color icell
x.colormapgui() # pops up the colormap GUI
x.animate() #pops up the animation GUI
x.grid(dim10,dim11,...,dimn0,dimn1) #:sets the dims(1..n) values for "out of graphics method" dime
x.resetgrid() # undo grid()
x.setcontinentstype(n)
x.destroy()
x.viewport = [0,1,0,1]
x.worldcoordinates = [x1,x2,y1,y2]
```

GUI

```

x.colormapgui()
x.animate()
x.projectiongui()
x.graphicsmethodgui()

```

Output

```

x.printer(prинтерname,['p'/'l']) # p/l: portrait/landscape
x.postscript(filename[.ps],['p'/'l']) # p/l: portrait/landscape
x.eps(filename[.ps],['p'/'l']) #p/l: portrait/landscape
x.pdf(filename[.ps],['p'/'l']) # p/l: portrait/landscape
x.gif(filename[.gif],['a'/'r'], ['p'/'l']) #a/r:append/replace
x.cgm(filename[.cgm],['a'/'r']) #a/r: append/replace
x.raster(filename[.ras],['a'/'r']) #a/r: append/replace
x.pstogif(filename[.ps],['l'/'p']) #p/l: portrait/landscape
x.backing_store()
x.showbg() # Show picture if plots we bg=1

```

Querying

(x=vcs.init(), gm is a graphics method)

```

gm.list() #list the option for a vcs object and the values they're set to
x.isgraphicsmethod(gm) # 1 if it is a graphicsmethod, 0 if not
x.is"vcsobjecttype"(gm) #1/0 "vcsobjecttype" can be any of
    # the graphics method ('boxfill',...)
    # the secondary methods ('marker'...)
    # template
x.show('vcsobjecttype') #print to screen all the vcs objects of type "vcsobjecttype" available
x.listelements({'vcsobjecttype'}) #returns a list of all the vcs objects of type "vcsobjecttype" a
x.orientation() # 'landscape'/'portrait'
x.islandscape() #1/0
x.isportrait() #1/0
x.canvasid()
x.canvasinfo()#{'width':w,'depth':d,'mapstate': m, 'height': h}
x.iscanvasdisplayed() #1/0
x.match_color(rgb/name,colormap='default')# Returns the color number that is closer from the requ

```

Help (x=vcs.init(), gm is a graphics method)

```

x.objecthelp(vcs object) #print the help for vcs object
vcs.help('function') # print the help for the vcs function
gm.list() #list the option for a vcs object and the values they're set to.
anypyobject.__doc__ #Any python object built in documentation

```

Scripting

```

x.scriptrun('name of vcs script') #reads a vcs script
x.scriptsav('name',['a/w']) #save the state of the system a/w: append (def)/replace
x.scriptobject(vcs object,'name[.scr]','a/w') #save the object as a python script unless
                                                # .scr specified, then saved as a vcs script, a/w:
x.saveinitialfile() #resave the VCS initial.attribute file
vcsobject.script ('file[.scr]','a/w') #save the vcs object to file, a/w:append (def)/replace
x.isinfile(graphicmethodobject)

```

Graphics Methods (x=vcs.init())
Creating/Retrieving/Deleting

```

x.create"graphicsmethod"('name','name to copy from')
x.get"graphicsmethod"('name')

```

```
x.removeobject (object)
```

Common Attributes

```
name = 'name'  
projection = projection # see next section  
x[y]ticlabels1[2] = [list]or {dic}  
x[y]mtics1[2] = [list] or {dic}  
datawc_x[y]1[2] = value # World coordinates  
x[y]axisconvert = 'linear', 'log10', 'ln', 'exp', or 'area_wt'
```

Specific Attributes

isofill/boxfill/meshfill

```
levels = [list] or (tuple) # ex [10,20,30] or [(20,30),(50,60)]  
fillareacolors = [list] or (tuple) # 0 <= values <= 255  
fillareastyle = 'solid', 'hatch', or 'pattern' # Not for Boxfill  
fillareaindices = # 1 <= value <= 17 or fillarea objects # Not for Boxfill  
legend = [list], (tuple) or {dictionary}  
ext_1 = 'n' / 'y' # Left arrow (off/on)  
ext_2 = 'n' / 'y' # Right arrow (off/on)  
missing = 0 <= value <= 255 # color of missing value
```

boxfill

```
level_1 = value # Plot data greater than level_1  
level_2 = value # Plot data lower than level_2  
color_1 = 0 <= value <= 255 # First color in palette  
color_2 = 0 <= value <= 255 # last color in palette  
boxfill_type = 'linear', 'log10' or 'custom'
```

meshfill

```
mesh = 'n' or 'y' # Show the mesh  
wrap = None or [Y,X] # Wrapping value on [Y,X]
```

isoline

```
level = [list] or (tuple) # ex [10,20,30] or [(20,30),(50,60)]  
label = 'n','y' or 0,1  
line = line object or value in line.type (0<=value<=4)  
linecolors =[list] or (tuple) 0 <= values <= 255  
text = 0<=values<=9 or texttable or textorientation object  
textcolors = [values] or from the objects passed to text
```

vector

```
line = line object or value in line.type (0<=value<=4)  
linecolor =[list] or (tuple) 0 <= values <= 255  
scale = value  
alignement = "head", "center", "tail" (or 0, 1, 2)  
type = "arrows", "barbs", "solidarrows" (or 0, 1, 2)  
reference = value
```

continents

```
line = line object or value in line.type (0<=value<=4)  
linecolor =[list] or (tuple) 0 <= values <= 255
```

```

type = 0 <= value <= 7 (or more if you added continents outline)
scatter (=xvxy with linecolor=whitecolor)
marker = None or values in marker.type (1<=value<=17)
markercolor =[list] or (tuple) 0 <= values <= 255
markersize = 1 <= value <= 300

```

xvxy/xvysy/yxvxsx

```

line = line object or value in line.type (0<=value<=4)
linecolors =[list] or (tuple) 0 <= values <= 255
marker = None or values in marker.type (1<=value<=17)
markercolor =[list] or (tuple) 0 <= values <= 255
markersize = 1 <= value<= 300

```

outfill

```

fillareacolor = [list] or (tuple) 0 <= values <= 255
fillareastyle = 'solid', 'hatch', or 'pattern'
fillareaindex = 1 <= value <= 17 or fillarea objects
outfill = values

```

outline

```

line = line object or value in line.type (0<=value<=4)
linecolor =[list] or (tuple) 0 <= values <= 255
outline = values

```

taylordiagram

```

detail = 75 # Graphic detail
max = None # Maximum value of standard deviation arc
quadrans = 1 # Number of quadrants: 1 or 2
skillValues = list # Skills levels to draw
skillColor = string or number (e.g. 'red' or 242)
skillDrawLabels = 'y' or 'n'
skillCoefficient = [1.0, 1.0, 1.0] # Used by skill function
referencevalue = 1.0
arrowlength = 0.05
arrowangle = 20.0
arrowbase = 0.75
xticlabels1 = *
xmtics1 = *
yticlabels1 = *
ymtics1 = *
cticlabels1 = *
cmtics1 = *
Marker
    status = []
    line = []
    id = []
    id_size = []
    id_color = []
    id_font = []
    symbol = []
    color = []
    size = []
    xoffset = []
    yoffset = []
    line_color = []
    line_size = []
    line_type = []

```

Template

```
name = `name'
file
priority = [value]
x = [value]
y = [value]
texttable = [string or texttable obj]
textorientation = [string or texorientation obj]
source/title/units ...same...
mean/min/max ...same...
comment1[2/3/4] ...same...
function/logicalmask/dataname ...same...
crdate/crttime/transformation ...same...
x[y/z/t]name[units/value/ ...same...
x[/y][min/max]tic1[/2]
priority = [value]
y[/x]1 = [value]
y[/x]2 = [value]
line = [string or line obj]
x[/y]label1[/2]
priority = [value]
y[/x] = [value]
line = [string or line obj]
data
priority = [value]
x1 = [value]
y1 = [value]
x2 = [value]
y2 = [value]
line = [string or line obj]
legend ...same...
line1[/2/3/4] ...same...
box1[2/3/4] ...same...
```

Secondary Methods

ALL (except projection)

```
name = 'name'
projection = projectionobject or 'projectionname' (see next page)
viewport = [0,1,0,1] # in % of page
worldcoordinate = [0,1,01] # [x1,X2,Y1,Y2]
```

textrientation (To)

```
height = 1 <= value <= 100
angle = 0 <= value<= 360
path = 'right','left','up','down' or (0,1,2,3)
halign = 'left','center','right' or (0,1,2)
valign = 'top','cap','half','base','bottom' or (0,1,2,3,4)
```

texttable (Tt)

```
font = 1 <= value <= 9
spacing = -50 <= value <= 50
expansion = 50 <= value <= 150
color = 0 <= value <= 255
```

line (Tl)

```
type = 'solid', 'dash', 'dot', 'dash-dot', 'long-dash' or (0, 1, 2, 3, 4)
width = 1 <= value <= 300
color = 0 <= value <= 255
```

marker (Tm)

```
type = 'dot', 'plus', 'star', 'circle', 'cross',
      'diamond', 'triangle_up', 'triangle_down', 'triangle_left',
      'triangle_right', 'square', 'diamond_fill', 'triangle_up_fill',
      'triangle_down_fill', 'triangle_left_fill', 'triangle_right_fill',
      'square_fill', or 0 <= value <= 17
width = 1 <= value <= 300
color = 0 <= value <= 255
```

fillarea(Tf)

```
name = 'name'
color = [list] or (tuple) 0 <= values <= 255
style = 'solid', 'hatch', or 'pattern'
index = 1 <= value <= 17
```

Projections (Proj)

```
name = 'name' # reserved names: 'linear', 'moleweide', 'robinson', 'polar'
type = 0<number<30, or type name
parameters = 15 paramters list #, 1.e20 value = automatic
or set an individual parameter name as listed bellow with itsdefault value
```

type = 1/2 ; utm/state plane

Not Implemented

type = 3 ; albers equal area

smajor = 1e+20

sminor = 1e+20

standardparallel1 = 1e+20

standardparallel2 = 1e+20

centralmeridian = 1e+20

originlatitude = 1e+20

falseeasting = 1e+20

falsenorthing = 1e+20

type=4 lambert conformal c

smajor = 1e+20

sminor = 1e+20

standardparallel1 = 1e+20

standardparallel2 = 1e+20

centralmeridian = 1e+20

originlatitude = 1e+20

falseeasting = 1e+20

falsenorthing = 1e+20

type 5 , mercator

smajor = 1e+20

sminor = 1e+20

centralmeridian = 1e+20

truescale = 1e+20

falseeasting = 1e+20

falsenorthing = 1e+20

type 6, polar stereographic

smajor = 1e+20

sminor = 1e+20

centerlongitude = 1e+20

truescale = 1e+20

falseeasting = 1e+20

falsenorthing = 1e+20

type 7, polyconic

smajor = 1e+20

sminor = 1e+20

centralmeridian = 1e+20

originlatitude = 1e+20

falseeasting = 1e+20

falsenorthing = 1e+20

type 8, equid conic

subtype = 0

smaior = 1e+20

sminor = 1e+20

centralmeridian = 1e+20

originlatitude = 1e+20

falseeast = 1e+20

falsenorthing = 1e+20

standardparallel = 1e+20

type 9, transverse mercator

smaior = 1e+20

sminor = 1e+20

factor = 1e+20

centralmeridian = 1e+20

originlatitude = 1e+20

falseeast = 1e+20

falsenorthing = 1e+20

type 10, stereographic

sphere = 1e+20

centerlongitude = 1e+20

centerlatitude = 1e+20

falseeast = 1e+20

falsenorthing = 1e+20

type 11, lambert azimuthal

sphere = 1e+20

centerlongitude = 1e+20

centerlatitude = 1e+20

falseeast = 1e+20

falsenorthing = 1e+20

type 12, azimuthal

sphere = 1e+20

centerlongitude = 1e+20

centerlatitude = 1e+20

falseeast = 1e+20

falsenorthing = 1e+20

type 13 gnomonic

sphere = 1e+20

centerlongitude = 1e+20

centerlatitude = 1e+20

falseeast = 1e+20

falsenorthing = 1e+20

type 14, orthographic

sphere = 1e+20

centerlongitude = 1e+20

centerlatitude = 1e+20

falseeast = 1e+20

falsenorthing = 1e+20

type 15, gen. vert. near per

sphere = 1e+20

height = 1e+20

centerlongitude = 1e+20

centerlatitude = 1e+20

falseeasting = 1e+20

falsenorthing = 1e+20

type 16, sinusoidal

sphere = 1e+20

centralmeridian = 1e+20

falseeasting = 1e+20

falsenorthing = 1e+20

type 17, equirectangular

sphere = 1e+20

centralmeridian = 1e+20

truescale = 1e+20

falseeasting = 1e+20

falsenorthing = 1e+20

type 18, miller cylindrical

sphere = 1e+20

centralmeridian = 1e+20

falseeasting = 1e+20

falsenorthing = 1e+20

type 19, van der griten

sphere = 1e+20

centralmeridian = 1e+20

originlatitude = 1e+20

falseeasting = 1e+20

falsenorthing = 1e+20

type 20, hotin oblique merc

subtype = 0
smajor = 1e+20
sminor = 1e+20
factor = 1e+20
originlatitude = 1e+20
falseeasting = 1e+20
falsenorthing = 1e+20
longitude1 = 0
latitude1 = 1e+20
longitude2 = 1e+20
latitude2 = 1e+20

type 21, space oblique merc

robinson

sphere = 1e+20
centralmeridian = 1e+20
falseeasting = 1e+20
falsenorthing = 1e+20
type 21, subtype = 0
smajor = 1e+20
sminor = 1e+20
falseeasting = 1e+20
falsenorthing = 1e+20
orbitinclination = 1e+20
orbitlongitude = 1e+20
satelliterevolutionperiod = 0
landsatcompensationratio = 1e+20

pathflag = 1e+20

type 23,alaska conformal

smajor = 1e+20

sminor = 1e+20

falseeast = 1e+20

falsenorthing = 1e+20

type 24, interrupted goode

sphere = 1e+20

type 25, Interrupt mollweide

sphere = 1e+20

centralmeridian = 1e+20

falseeast = 1e+20

falsenorthing = 1e+20

type 27, hammer

sphere = 1e+20

centralmeridian = 1e+20

falseeast = 1e+20

falsenorthing = 1e+20

type 28/29,wagner iv/vii

sphere = 1e+20

centralmeridian = 1e+20

falseeast = 1e+20

falsenorthing = 1e+20

type 30, oblated equal area

sphere = 1e+20

shapem = 1e+20

shapen = 1e+20

centerlongitude = 1e+20

centerlatitude = 1e+20

falseeast = 1e+20

falsenorthing = 1e+20